```python
# -*- coding: utf-8 -*-
"""
Created on Tue Mar  5 10:48:44 2019

@author: Nate
"""

import numpy as np
import matplotlib.pyplot as plt
import pdb

#fig2, axes2 = plt.subplots()


################################################################
################################################################
#defining constants

#number of trials
N = 100000


#α=0.2 to 2 in increments of 0.2

alpha = [i*.2 for i in range(1,11)]
rmax_list = [(3.2/alph) for alph in alpha]
r0 = np.array([0.01,0,0])
energy = []
std_dev = []



#sigma = [4*i for i in alpha]
#rmax = 1/(2*alpha)




################################################################
################################################################
#defining functions


def get_delta_r(rmax):
    delta_r = np.array([rmax*(np.random.uniform()-.5),rmax*(np.random.uniform()-.5),rmax*(np.random.uniform()-.5)])
    return(delta_r)

#evaluate ratio
def evaluate_ratio(r, delta_r):
    global count
    A = np.exp(-2*alpha[i]*np.linalg.norm(r))
    B = np.exp(-2*alpha[i]*np.linalg.norm(r+delta_r))
    ratio = B/A
    
    
    if ratio > np.random.uniform():
        count+=1
        return(True)
    else:
        return(False)

def generate_point(r, delta_r):
    if evaluate_ratio(r, delta_r) == True:
        return(r+delta_r)
    else:
```

```
        return(r)


#################################################################
#################################################################



for i in range(len(alpha)):

    #generating points
    r = np.array([r0])
    count = 0
    en = 0

    for j in range(N):
        #r2=np.random.exponential(.5/alpha[i])
        #en +=-alpha[i]**2 + alpha[i]/np.linalg.norm(r2) - 1/np.linalg.norm(r2)

        #pdb.set_trace()
        r = np.append(r, [generate_point(r[len(r)-1], get_delta_r(rmax_list[i]))], axis
        = 0)
        en +=-(alpha[i]**2)/2 + alpha[i]/np.linalg.norm(r[j]) - 1/np.linalg.norm(r[j])


    print(rmax_list[i], 'accepted: ',count, 'rejected: ',N-count)
    print('Acceptance Ratio: ', count/N)
    energy.append(en/N)


    to_sum = 0
    for j in range(N):
        to_sum += (-(alpha[i]**2)/2 + alpha[i]/np.linalg.norm(r[j]) -
        1/np.linalg.norm(r[j])-energy[i])**2

    std_dev.append(np.sqrt(to_sum)/np.sqrt(N))

    '''
    if i == 1:
        axes2.hist([np.linalg.norm(i) for i in r], bins = 100, normed = True)
        axes2.plot([n*.1 for n in range(0,3000)]    ,
        [2*alpha[i]*np.e**(-2*alpha[i]*n*.1)  for n in range(0,3000)])
'''


print(energy)

#################################################################
#################################################################
#Plotting


fig1, axes1 = plt.subplots()

#axes1.scatter(alpha, [3/2*num-2**(3/2)/(np.pi**.5)*num**.5 for num in alpha])
#axes1.scatter(8/(9*np.pi), 3/2*(8/(9*np.pi))-2**(3/2)/(np.pi**.5)*(8/(9*np.pi))**.5)
#axes1.plot(alpha, energy)
axes1.errorbar(alpha, energy, yerr=std_dev, fmt='o', linestyle = '-')
axes1.plot(alpha, [(num**2)/2-num for num in alpha])

axes1.set_ylabel('Energy')
axes1.set_xlabel('alpha')
axes1.set_title("Energy vs alpha", va='bottom')


plt.show()
```